अवधारणा · 23 मई 2026

← पिछला·अगला →

# वे 24 शब्द: क्रिप्टोग्राफिक पहचान क्या है

एक क्रिप्टोग्राफिक पहचान पासवर्ड नहीं है: इसे कोई सर्वर स्टोर नहीं करता है और इसे पुनर्प्राप्त (recover) नहीं किया जा सकता है। BIP39 तंत्र का एक शैक्षिक विवरण, वास्तव में चौबीस शब्द क्यों, और उन्हें रखने वाले पर वास्तविक जिम्मेदारी क्या है।

**बात समझने के लिए:** यदि आप अपना Gmail पासवर्ड भूल जाते हैं, तो Google इसे आपके लिए रीसेट कर देता है। यदि आप उन 24 शब्दों को खो देते हैं जो एक क्रिप्टोग्राफिक पहचान बनाते हैं, तो उन्हें मांगने के लिए कोई नहीं है। ऐसा नहीं है कि प्रक्रिया सख्त है - बात यह है कि दूसरी तरफ कोई है ही नहीं। वह अंतर ही सबसे बड़ा अंतर है।

## पासवर्ड और पहचान के बीच का अंतर

इंटरनेट के क्लासिक मॉडल में, पासवर्ड उपयोगकर्ता की पहचान नहीं है। यह एक प्रमाण (voucher) है। उपयोगकर्ता की एक पहचान होती है — एक नाम, एक ईमेल, एक ग्राहक नंबर — और सर्वर के सामने यह साबित करने के लिए कि वह वही है जो वह कहता है, वह एक पासवर्ड प्रस्तुत करता है जिसकी सर्वर अपने पास संग्रहीत छाप (footprint) से तुलना करता है। यदि छापें मेल खाती हैं, तो सर्वर सत्र की अनुमति देता है। यदि पासवर्ड खो जाता है, तो उपयोगकर्ता वही उपयोगकर्ता रहता है; वह जो खोता है वह प्रमाण है, और इसे बहाल करने के लिए एक पुनर्प्राप्ति प्रक्रिया — पंजीकृत पते पर एक ईमेल, एक सुरक्षा प्रश्न — मौजूद होती है।

क्रिप्टोग्राफिक पहचान अलग तरह से काम करती है। यह कोई क्रेडेंशियल नहीं है जिसे कोई संग्रहीत छाप से तुलना करे; यह अपने आप में एक पूर्ण गणितीय रहस्य *है*। इससे कोई फर्क नहीं पड़ता कि यह कहाँ रहता है — कागज पर, किसी डिवाइस में, यहाँ तक कि किसी बाहरी सर्वर पर — पहचान अपने गणित के कारण मौजूद है, न कि उसे सत्यापित करने वाले के कारण। यहाँ «SHA-256 वास्तव में क्या है» में हमने जो गुण देखा था, उसके समान एक गुण दिखाई देता है: रहस्य को दिखाकर स्वामित्व साबित नहीं किया जाता है, बल्कि हस्ताक्षर करने के लिए इसका उपयोग करके साबित किया जाता है। इस तरह से उत्पन्न हस्ताक्षर को कोई भी एक सार्वजनिक मूल्य (public value) के साथ जांच सकता है जो गणितीय रूप से रहस्य से ही प्राप्त होता है, रहस्य को जानने की आवश्यकता के बिना, और जांच में किसी तीसरे पक्ष की मध्यस्थता के बिना। जिसके पास रहस्य है, वही पहचान है; जो इसे खो देता है, उसकी वह पहचान समाप्त हो जाती है। निर्णय स्पष्ट है: **पहचान वापस करने के लिए मांगने के लिए कोई नहीं है। वह व्यक्ति अस्तित्व में नहीं है, क्योंकि उसके पास वह पहले स्थान पर थी ही नहीं।**

## चौबीस शब्द क्या दर्शाते हैं

क्रिप्टोग्राफिक पहचान को आमतौर पर बत्तीस बाइट्स — दो सौ छप्पन बिट्स — के गणितीय रहस्य द्वारा दर्शाया जाता है। एक ऐसी संख्या जिसे याद रखना कठिन है और बिना किसी त्रुटि के लिखना और भी कठिन है। क्रिप्टोग्राफ़िक उद्योग ने इस समस्या को 2013 में BIP39 नामक एक छोटे और सुरुचिपूर्ण मानक के साथ हल किया: उन दो सौ छप्पन बिट्स को दो हजार अड़तालिस शब्दों की आधिकारिक सूची से लिए गए चौबीस शब्दों के अनुक्रम के रूप में प्रदर्शित करने का एक तरीका। इसके पीछे का अंकगणित खूबसूरती से फिट बैठता है; जो कोई भी इसे विस्तार से देखना चाहता है वह इसे हाशिए (aside) में पा सकता है।

गिनती अंत से शुरू होती है। हम आठ बिट्स चेकसम जोड़कर रहस्य के दो सौ छप्पन बिट्स का प्रतिनिधित्व करना चाहते हैं: कुल दो सौ चौंसठ बिट्स। यदि हम उन्हें चौबीस शब्दों में वितरित करते हैं — बिना किसी नुकसान के नोट करने और बोलकर लिखाने के लिए एक प्रबंधनीय संख्या — तो प्रत्येक शब्द को ठीक ग्यारह बिट्स की जानकारी प्रदान करनी होगी। और ग्यारह बिट्स दो की घात

ग्यारह संभावनाओं के बराबर हैं, यानी दो हजार अड़तालिस। इसीलिए आधिकारिक BIP39 शब्दावली का आकार बिल्कुल इतना ही है: सूची समस्या के अनुसार मौजूद है, न कि इसके विपरीत।

गिनती दिखावटी नहीं है। यदि कोई तेईस शब्दों को सही ढंग से लिखता है और चौबीसवें शब्द में गलती करता है, तो चेकसम इसका पता लगा लेगा: सॉफ्टवेयर उसे बताएगा "यह अनुक्रम मान्य नहीं है"। यदि कोई चौबीसों शब्दों को सही लिखता है, तो सॉफ्टवेयर बिना किसी अस्पष्टता के वही पहचान प्राप्त कर लेगा। शब्दों की सूची का चुनाव भी सोच-समझकर किया गया है: BIP39 शब्दावली के शब्द छोटे हैं, एक-दूसरे से अलग हैं, बिना किसी विशेष उच्चारण चिह्नों के हैं, जिन्हें ध्वन्यात्मक और वर्तनी संबंधी भ्रमों को कम करने के लिए चुना गया है। यह एक ऐसी शब्दावली है जिसे मनुष्यों द्वारा बिना किसी नुकसान के याद रखने, लिखने और बोलकर लिखाने के लिए डिज़ाइन किया गया है।

# वाक्यांश से कुंजी तक

वे चौबीस शब्द वे क्रिप्टोग्राफ़िक कुंजी नहीं हैं जो संदेशों पर हस्ताक्षर करती हैं। वे मूल एन्ट्रोपी का एक पुनर्प्राप्त करने योग्य प्रतिनिधित्व हैं, जो PBKDF2 नामक एक नियतात्मक प्रक्रिया के माध्यम से चौंसठ-बाइट बीज (seed) में बदल जाते हैं। उसी बीज से, नियतात्मक रूप से, वे विशिष्ट क्रिप्टोग्राफ़िक कुंजियाँ प्राप्त होती हैं जिनका उपयोग उपयोगकर्ता करता है: हस्ताक्षर करने के लिए एक निजी कुंजी और एक संबंधित सार्वजनिक कुंजी जिसे हस्ताक्षरों को सत्यापित करने के लिए प्रकाशित किया जाता है। विभिन्न प्रणालियों में एक ही तंत्र: क्रिप्टोकरेंसी secp256k1 वक्र का उपयोग करती है; Signal प्रोटोकॉल और कई आधुनिक प्रणालियाँ Curve25519 वक्र पर Ed25519 का उपयोग करती हैं। Ed25519 जैसे विशिष्ट वक्र के लिए, BIP32 और SLIP-0010 मानक उस चौंसठ-बाइट बीज को लेते हैं और नियतात्मक रूप से बत्तीस बाइट्स प्राप्त करते हैं जो प्रभावी हस्ताक्षर कुंजी का गठन करते हैं — वही बत्तीस बाइट्स जिनसे अगले अनुभाग का कोड उदाहरण शुरू होता है।

यह वह मानक तरीका है जिससे पूरा उद्योग उपयोगकर्ता को तंत्र प्रस्तुत करता है —क्रिप्टोकरेंसी वॉलेट, विकेंद्रीकृत पहचान प्रबंधक, अपने स्थायी पहचान भाग में Signal, उनके बीच Solo2—: उपयोगकर्ता, व्यवहार में, बीज या व्युत्पन्न कुंजियों को कभी नहीं देखता है। वह अपनी पहचान बनाते समय चौबीस शब्दों को देखता है और वैकल्पिक रूप से उन्हें एक कागज़ पर नोट कर लेता है। फिर शब्द उसके उपकरणों के बीच यात्रा करते हैं जब वह पहचान को स्थानांतरित करना चाहता है: वह उन्हें नए एप्लिकेशन में दर्ज करता है, एप्लिकेशन वही बीज, वही कुंजियाँ, वही पहचान प्राप्त करता है। यह एक पोर्टेबल, क्रिप्टोग्राफ़िक रूप से ठोस और उचित सीमाओं के भीतर याद रखने योग्य तंत्र है।

# कुंजी के साथ हस्ताक्षर कैसे करें (Zig की एक झलक)

Zig में, एक बार जब आपके पास चौबीस शब्दों से प्राप्त बत्तीस-बाइट बीज हो जाता है, तो Ed25519 के साथ एक संदेश पर हस्ताक्षर करना कुछ ही पंक्तियों में आ जाता है:

```
const std = @import("std");
const Ed25519 = std.crypto.sign.Ed25519;

// 'semilla' son los 32 bytes derivados de las 24 palabras.
const par = Ed25519.KeyPair.create(semilla);

// Firmar un mensaje con la clave privada:
const mensaje = "Este mensaje lo escribí yo.";
const firma = try par.sign(mensaje, null);

// Cualquiera con la clave pública del par puede verificar:
try Ed25519.Signature.verify(firma, mensaje, par.public_key);
```

हस्ताक्षर करने की क्रिया चौंसठ बाइट्स उत्पन्न करती है —जिन्हें हस्ताक्षर कहा जाता है— जिन्हें केवल संबंधित निजी कुंजी से ही उत्पन्न किया जा सकता था। सत्यापन सार्वजनिक है: सार्वजनिक कुंजी वाला कोई भी व्यक्ति यह जाँच सकता है कि हस्ताक्षर संदेश

से मेल खाता है या नहीं। निजी कुंजी के बिना, कोई भी उस संदेश के लिए वैध हस्ताक्षर तैयार नहीं कर सकता है; सार्वजनिक कुंजी के साथ, हर कोई यह पता लगा सकता है कि हस्ताक्षर वैध है या नहीं। यह विषमता ही हस्ताक्षरकर्ता को रहस्य साझा किए बिना लेखकत्व प्रदर्शित करने की अनुमति देती है।

पिछला उदाहरण नियमावली का न्यूनतम संस्करण है। Solo2 के वास्तविक कोड में, श्रृंखला दो फाइलों से होकर गुजरती है, एक JavaScript में जो उपयोगकर्ता के ब्राउज़र में रहती है और चौबीस शब्दों से एंट्रॉपी का पुनर्निर्माण करती है, दूसरी Zig में *zcatcrypto* लाइब्रेरी के भीतर जो उस एंट्रॉपी को लेती है और विशिष्ट क्रिप्टोग्राफिक कुंजियाँ प्राप्त करती है। ब्राउज़र की ओर से शुरू करते हुए:

```
// solo2/web-app/js/lib/bip39.js
async function mnemonicToEntropy(mnemonic, lang) {
    const validation = await validateMnemonic(mnemonic, lang);
    if (!validation.valid) {
        return { entropy: null, valid: false, error: validation.error };
    }
    const wordlist = WORDLISTS[lang || 'en'];
    const words = mnemonic.trim().split(/\s+/);

    // Cada palabra aporta 11 bits (su índice en la lista de 2048).
    let bits = '';
    for (let i = 0; i < words.length; i++) {
        bits += wordlist.indexOf(words[i]).toString(2).padStart(11, '0');
    }

    // 24 palabras = 264 bits. Los primeros 256 son la entropía.
    const entropyBytes = new Uint8Array(32);
    for (let j = 0; j < 32; j++) {
        entropyBytes[j] = parseInt(bits.slice(j * 8, (j + 1) * 8), 2);
    }
    return { entropy: entropyBytes, valid: true };
}
```

वे बत्तीस बाइट्स एंट्रॉपी, उसी चरण में प्राप्त अन्य बत्तीस के साथ, Zig के WebAssembly मॉड्यूल की यात्रा करती हैं जो वास्तविक Ed25519 कुंजियाँ उत्पन्न करता है। पूरा फंक्शन, अपनी अंतिम मेमोरी क्लीनअप के साथ, एक स्क्रीन पर फिट बैठता है:

```
// zcatcrypto/wasm/bindings/identity.zig
const Ed25519 = std.crypto.sign.Ed25519;
const X25519  = std.crypto.dh.X25519;

export fn identity_generate() ?*IdentityHandle {
    var seed: [64]u8 = undefined;
    if (!common.getRandomBytes(&seed)) return null;

    const handle = common.wasm_allocator.create(IdentityHandle) catch return null;

    // Bytes 0..31: semilla determinista del par Ed25519 (firma).
    const sign_kp = Ed25519.KeyPair.generateDeterministic(seed[0..32].*) catch {
        common.wasm_allocator.destroy(handle);
        return null;
    };
    handle.sign_secret = sign_kp.secret_key.toBytes();
    handle.sign_public = sign_kp.public_key.toBytes();
```

```
    // Bytes 32..63: secreto X25519 (para acordar claves de cifrado con el otro).
    handle.exchange_secret = seed[32..64].*;
    handle.exchange_public = X25519.recoverPublicKey(handle.exchange_secret) catch {
        common.wasm_allocator.destroy(handle);
        return null;
    };

    @memset(&seed, 0);  // Borra la semilla de la memoria.
    return handle;
}
```

दो विवरण ध्यान देने योग्य हैं। पहला: एक ही सीड (seed) हमेशा एक ही कुंजी युग्म उत्पन्न करता है — ठीक यही वह चीज है जो नए डिवाइस में चौबीस शब्द दर्ज करके पहचान पुनर्प्राप्ति की अनुमति देती है। दूसरा: अंतिम पंक्ति में सीड को स्पष्ट रूप से मेमोरी से मिटा दिया जाता है। उस बिंदु के बाद, स्वयं फंक्शन भी कुंजियों का पुनर्निर्माण नहीं कर सका; उपयोगकर्ता के शब्द ही एकमात्र स्रोत होंगे।

**उन लोगों के लिए जो इसे छोटी संख्याओं के साथ जाँचना चाहते हैं।** हस्ताक्षर योजना को हाथ से गणना करने के लिए पर्याप्त छोटे अंकों के साथ पूरी तरह से समझा जा सकता है। जो लोग अंकगणित में नहीं जाना चाहते वे लेख के प्रवाह को खोए बिना इस ब्लॉक को छोड़ सकते हैं; जो लोग तंत्र को चरण-दर-चरण काम करते देखना चाहते हैं वे इसे यहाँ पाएंगे। **सार्वजनिक नियम**, जिन्हें कोई भी पढ़ सकता है: एक अभाज्य संख्या (prime) $p = 23$ (वास्तविक Ed25519 में यह लगभग सतहत्तर अंकों का होता है; हम तेईस का उपयोग करते हैं ताकि गणना एक पृष्ठ पर फिट हो सके), एक आधार $g = 2$ जिसका इस समूह में क्रम (order) $q = 11$ है, और यह परंपरा कि $g$ के साथ सारा अंकगणित $módulo\ p$ किया जाता है और सभी घात (exponents) $módulo\ q$ कम हो जाते हैं। **निजी चुनाव**, केवल एक और कभी साझा नहीं किया गया: रहस्य $x = 6$। यही पहचान है।

**चरण 1 — पहचान का सार्वजनिक हिस्सा।** इसकी गणना एक बार की जाती है और खुले तौर पर प्रकाशित की जाती है।

$y = g^x \bmod p$

$y = 2^6 \bmod 23 = 64 \bmod 23 = 18$

पहचान का सार्वजनिक हिस्सा **18** है। कोई भी इसे ले सकता है और इस पहचान के साथ किए गए हस्ताक्षरों को सत्यापित करने के लिए इसका उपयोग कर सकता है। कोई भी, केवल 18 को देखकर, रहस्य 6 को पुनर्प्राप्त नहीं कर सकता है: यह असतत लघुगणक (discrete logarithm) की समस्या है जिस पर हम अंत में लौटेंगे।

**चरण 2 — एक संदेश पर हस्ताक्षर करना।** पहचान धारक संदेश $m = 7$ पर हस्ताक्षर करना चाहता है। वह एक नया यादृच्छिक मान $k = 4$ चुनकर शुरू करता है, जिसका उपयोग केवल एक बार किया जाएगा और कभी साझा नहीं किया जाएगा (वास्तविक Ed25519 में, $k$ को संदेश और रहस्य से नियतात्मक रूप से प्राप्त किया जाता है ताकि पुन: उपयोग के खतरे से बचा जा सके, लेकिन इसकी भूमिका बिल्कुल यही है)। फिर वह तीन संख्याओं की गणना करता है:

$r = g^k \bmod p = 2^4 \bmod 23 = 16$

$e = H(r, m) \bmod q = (16 + 7) \bmod 11 = 1$

$s = (k + x \cdot e) \bmod q = (4 + 6 \cdot 1) \bmod 11 = 10$

हस्ताक्षर युग्म **(r, s) = (16, 10)** है। यह संदेश के साथ खुले में यात्रा करता है। कोई भी इसे पढ़ सकता है। उपदेशात्मक नोट: वास्तविक Ed25519 में फंक्शन $H$ SHA-512 है, जो क्रिप्टोग्राफिक रूप से मजबूत है; यहाँ हम सरलीकरण $e = (r + m)\ mod\ q$ का उपयोग करते हैं ताकि पाठक हैश की गणना किए बिना चरणों का पालन कर सके। एल्गोरिथ्म की संरचना वही है।

**चरण 3 — हस्ताक्षर सत्यापित करना।** सत्यापनकर्ता के पास सार्वजनिक हिस्सा $y = 18$, संदेश $m = 7$, और हस्ताक्षर $(r, s) = (16, 10)$ है। वह उसी तरह $e$ का पुनर्निर्माण करता है — $e = (16 + 7) \bmod 11 = 1$ — और जाँचता है कि क्या यह समानता

सत्य है:

$g^s \bmod p \stackrel{?}{=} r \cdot y^e \bmod p$

दोनों पक्षों की अलग-अलग गणना करता है:

Izquierda: $2^{10} \bmod 23 = 1024 \bmod 23 = 12$

Derecha: $16 \cdot 18^1 \bmod 23 = 288 \bmod 23 = 12$

दोनों पक्ष **12** देते हैं। हस्ताक्षर मान्य है। सार्वजनिक हिस्सा 18 रखने वाला कोई भी इस निष्कर्ष पर पहुँच सकता है बिना यह जाने कि रहस्य 6 था।

**और कोई तीसरा जो जालसाजी की कोशिश करे?** ईवा ने चैनल से गुजरने वाली सभी सार्वजनिक चीजें देखी हैं: $p$ = 23, $g$ = 2, $q$ = 11, $y$ = 18, $m$ = 7, $r$ = 16, $s$ = 10। इस पहचान के नाम पर एक *अलग* संदेश पर हस्ताक्षर करने के लिए, उसे $x$ जानने की आवश्यकता होगी। उसका एकमात्र तरीका खुद से पूछना है: «किस घात (exponent) $x$ के लिए $2^x \bmod 23 = 18$ सत्य है?». $p$ = 23 के साथ वह 0, 1, 2, 3, ... आजमा सकती है और इसे सेकंडों में ढूंढ सकती है। लेकिन 23 को Ed25519 के वास्तविक आयामों के अभाज्य संख्या से बदलने पर, संभावित घातों का स्थान दृश्यमान ब्रह्मांड के परमाणुओं की संख्या से अधिक हो जाता है। **आज मानवता को ज्ञात ऐसा कोई एल्गोरिथ्म नहीं है जो अरबों वर्षों से कम समय में उस स्थान को पार कर सके।** यह वही असतत लघुगणक की समस्या है जो पिछले लेख के Diffie-Hellman का आधार है, यहाँ हस्ताक्षर योजना पर लागू की गई है।

हमने अभी जो कुछ भी देखा है वह *बिल्कुल* Schnorr है, वह हस्ताक्षर योजना जिसका Ed25519 एक दीर्घवृत्तीय वक्र (elliptic curve) के लिए अनुकूलित संस्करण है। वास्तविक Ed25519 में, सभी ऑपरेशन एक अभाज्य संख्या के मॉड्यूलस पूर्णांकों के बजाय एक विशिष्ट वक्र (Curve25519) के बिंदुओं पर किए जाते हैं, और फंक्शन $H$ हमारे द्वारा ऊपर उपयोग किए गए खिलौना योग के बजाय SHA-512 है। दो प्रतिस्थापन कार्यान्वयन समायोजन हैं — ब्रूट फोर्स के लिए क्रिप्टोग्राफिक प्रतिरोध प्राप्त करना, $k$ के लिए अतिरिक्त सुरक्षा गुण प्राप्त करना। एल्गोरिथम संरचना, तीन ऑपरेशन, विषमता का कारण, वही हैं।

यहाँ एक संक्षिप्त विराम उचित है, क्योंकि पूरी श्रृंखला को एक त्वरित नज़र में तिकड़ी के एक अन्य प्रिमिटिव: हैश के साथ भ्रमित किया जा सकता है। यह वह नहीं है। हैश एक अनूठा फंक्शन है जो संपीड़ित करता है — कई बाइट्स अंदर जाते हैं, एक छोटा पदचिह्न बाहर आता है, वहीं रास्ता खत्म हो जाता है। एक क्रिप्टोग्राफिक पहचान एक पूरक गणितीय जोड़ी है: रहस्य रहता है और हस्ताक्षर करता है; इसका सार्वजनिक प्रतिरूप प्रकाशित और सत्यापित होता है। जहाँ हैश जानकारी को एक ही दिशा में ढहा देता है, पहचान दो हिस्सों के बीच एक विषमता स्थापित करती है। हैश प्रमाणित करता है कि क्या कहा गया था; पहचान प्रमाणित करती है कि किसने कहा था।

## वाक्यांश क्या नहीं है

तीन सामान्य भ्रांतियों को दूर करना उचित है। वाक्यांश अपने सही अर्थ में पासवर्ड नहीं है: इसकी तुलना सर्वर पर संग्रहीत फिंगरप्रिंट से नहीं की जाती है; पहचान को गणितीय रूप से पुनर्गठित करने के लिए इसे उपयोगकर्ता के डिवाइस में दर्ज किया जाता है। वाक्यांश को पुनर्प्राप्त नहीं किया जा सकता है: यदि यह खो जाता है, तो इसे मांगने के लिए कोई नहीं है; यदि इसे डुप्लिकेट किया जाता है, तो पहचान भी डुप्लिकेट हो जाती है। वाक्यांश पहचान से अलग किया जा सकने वाला क्रेडेंशियल नहीं है: वाक्यांश पहचान *है*। जिसके पास यह है वह उस पहचान के रूप में कार्य कर सकता है, बिना किसी अतिरिक्त अनुमति के, बिना प्राधिकरण प्रक्रिया के, बिना किसी संभावित पुनर्प्राप्ति के।

यह तीसरा गुण ही है जो मामले के भार को बदल देता है। खोया हुआ पासवर्ड एक प्रशासनिक परेशानी है। खोई हुई क्रिप्टोग्राफ़िक पहचान स्वयं पहचान है। तीसरे पक्ष द्वारा पाया गया वाक्यांश वाला कागज़ खाता चोरी का जोखिम नहीं है: यह पूरी पहचान को सौंपना है। सिस्टम का वादा —कि कोई भी आपकी पहचान को रद्द नहीं कर सकता है या आपको मनमाने ढंग से ब्लॉक नहीं कर सकता है— जिम्मेदारी के साथ अविभाज्य रूप से आता है —कि आप किसी ऐसी चीज़ के एकमात्र संरक्षक हैं जिसे आपके लिए कोई भी बहाल नहीं कर सकता है।

## वादा और भार

क्रिप्टोग्राफ़िक पहचान मॉडल को अक्सर *स्व-संप्रभु* —अंग्रेजी साहित्य में self-sovereign— कहा जाता है। शब्द का चुनाव जानबूझकर किया गया है और यह स्थिति का काफी सटीक वर्णन करता है। उपयोगकर्ता अपनी पहचान पर लगभग मध्यकालीन अर्थों में संप्रभु है: कोई भी राजा, कोई जारीकर्ता, कोई केंद्रीय प्राधिकरण इसे प्रदान नहीं करता है; न ही उपरोक्त में से कोई इसे वापस ले सकता है। लेकिन साथ ही, मध्यकालीन सम्राट की तरह, उपयोगकर्ता अपनी गलतियों का पूरा परिणाम भुगतता है: यदि वह मुहर खो देता है तो उसके स्थान पर निर्णय लेने के लिए कोई प्रतिनिधि नहीं होता है।

किसी तीसरे पक्ष द्वारा प्रबंधित पहचान और स्व-संप्रभु पहचान के बीच चयन का कोई एक सार्वभौमिक सही उत्तर नहीं है। किसी अप्रासंगिक मंच खाते के लिए, प्रबंधित पहचान शायद जोखिम के अनुपात में है। एक पेशेवर पहचान के लिए जो कानूनी रूप से बाध्यकारी दस्तावेजों पर हस्ताक्षर करती है, एक आर्थिक पहचान के लिए जो अपनी बचत की रक्षा करती है, उन ग्राहकों के साथ पेशेवर संचार की पहचान के लिए जिन्होंने संवेदनशील जानकारी सौंपी है, मामला बदल जाता है। वहाँ प्रश्न यह होना बंद हो जाता है कि «क्या यह सुविधाजनक है?» और यह बन जाता है कि «मेरे अलावा किसके पास मेरे रूप में कार्य करने की शक्ति है, और किन परिस्थितियों में?».

## वास्तविक प्रणालियों में यह तंत्र कहाँ दिखाई देता है

BIP39 का जन्म 2013 में Bitcoin की दुनिया में हुआ था और यह तेजी से पूरे क्रिप्टोकरेंसी इकोसिस्टम में फैल गया: आज कोई भी गंभीर वॉलेट अपने धारक की आर्थिक पहचान के बैकअप के रूप में बारह या चौबीस शब्दों के BIP39 वाक्यांश को स्वीकार करता है। क्रिप्टोकरेंसी के बाहर, वही अंतर्निहित अवधारणा — एक क्रिप्टोग्राफ़िक जोड़ी जो बिना किसी मध्यस्थ के लेखकत्व साबित करती है — अलग सिंटैक्स वाले अन्य सिस्टम में दिखाई देती है। SSH कुंजियाँ जो एक सिस्टम एडमिनिस्ट्रेटर अपने सर्वर तक पहुँचने के लिए उपयोग करता है, एक क्लासिक मामला है: एक निजी कुंजी जो एडमिनिस्ट्रेटर अपनी मशीन पर रखता है और एक सार्वजनिक कुंजी जिसे प्रत्येक सर्वर पर कॉपी किया जाता है; केंद्रीकृत सेवा के बराबर कोई भी इकाई हस्तक्षेप नहीं करती है। Signal प्रोटोकॉल डिवाइस पर स्थायी कुंजी सामग्री के साथ Ed25519 का उपयोग करता है; यूरोपीय eIDAS, अपने योग्य हस्ताक्षर भाग में, उसी क्रिप्टोग्राफ़िक सिद्धांत पर टिका है, इस अंतर के साथ कि कुंजी उपयोगकर्ता के बजाय एक योग्य ट्रस्ट सेवा प्रदाता द्वारा रखी जाती है।

Solo2, इस प्रकाशन का प्रकाशन मंच, प्रत्येक उपयोगकर्ता की पहचान के रूप में चौबीस शब्दों के BIP39 वाक्यांश का उपयोग करता है। उपयोगकर्ता, अपना खाता बनाते समय, शब्दों को एक बार देखता है। वे किसी भी Solo2 सर्वर या किसी और के सर्वर पर संग्रहीत नहीं होते हैं: यदि उपयोगकर्ता उन्हें नोट करता है और उनकी रक्षा करता है, तो वह अपनी पहचान हमेशा के लिए रखता है। यदि वह उन्हें खो देता है, तो वह उन्हें खो देता है। यह बीच में कोई ऑपरेटर नहीं होने वाली वास्तुकला का सुसंगत परिणाम है: यदि Solo2 उस उपयोगकर्ता को पहचान वापस कर सकता है जिसने इसे खो दिया है, तो वह इसे उसे भी दे सकता है जो इसे प्राप्त करने के लिए Solo2 पर दबाव डालता है।

## पेशेवर पाठक के लिए

पेशेवर संदर्भ में क्रिप्टोग्राफ़िक स्व-संप्रभु (autosoberana) पहचान अपनाने का मूल्यांकन करने वालों के लिए चार विचार:

1. वाक्यांश ही पहचान है। भौतिक सुरक्षा — कागज, विभिन्न स्थानों पर कई प्रतियां, अंततः दीर्घकालिक उपयोग के लिए उत्कीर्ण धातु — डिजिटल सुरक्षा की तुलना में अधिक गारंटी प्रदान करती है, जो नुकसान के जोखिम को कम किए बिना हमले की सतह को बढ़ाती है।
2. कोई रिकवरी नहीं है। इस धारणा के साथ प्रक्रिया को डिजाइन करना कि एक दिन प्राथमिक प्रति खो जाएगी, उस दिन इसे खोजने की तुलना में बहुत अधिक विवेकपूर्ण है जिस दिन यह खो जाती है। भौगोलिक रूप से अलग दूसरी प्रति लगभग सभी परिदृश्यों को हल करती है।
3. यह eIDAS योग्य प्रमाण पत्र के समान नहीं है। संघ में योग्य हस्ताक्षर के लिए — नोटरी विलेख, प्रशासन के साथ कुछ प्रक्रियाएं — कानून के लिए एक योग्य प्रदाता की आवश्यकता होती है जो कुंजी रखता है। क्रिप्टोग्राफ़िक स्व-संप्रभु पहचान

पेशेवर संचार और साक्ष्य मूल्य के साथ दस्तावेजी हस्ताक्षर के लिए काम करती है, लेकिन उन मामलों में स्वचालित रूप से योग्य प्रमाण पत्र की जगह नहीं लेती है जहां नियम इसकी आवश्यकता होती है।

4. यदि पहचान हस्तांतरित की जानी है — विरासत, पेशेवर उत्तराधिकार, गतिविधि का बंद होना — तो प्रक्रिया को बाद में नहीं, बल्कि पहले तैयार करना उचित है। लाख (lacre) से सील किए गए लिफाफों के साथ औपचारिक प्रक्रियाएं, एक वसीयत निष्पादक को निर्देश, नोटरी कार्यालय में जमा करना, क्लासिक व्यवस्थाएं हैं जो संपत्ति की क्रिप्टोग्राफ़िक प्रकृति के साथ पूरी तरह से संगत हैं।

---

*यह लेख उस वैचारिक तिकड़ी को समाप्त करता है जिसने चक्र की शुरुआत की थी — hash, एन्क्रिप्शन, पहचान —। तीनों विचार एक-दूसरे पर बने हैं: hash अपरिवर्तनीय फिंगरप्रिंट देता है, एन्क्रिप्शन विश्वसनीय तीसरे पक्ष के बिना गोपनीयता देता है, पहचान प्रदान करने वाले तीसरे पक्ष के बिना लेखकत्व देती है। तीनों एक ऐसी विशेषता साझा करते हैं जो वैचारिक भी नहीं है: वे तकनीकी क्षमताओं को स्थानांतरित करते हैं, जो पारंपरिक रूप से ऑपरेटर के पास होती थीं, सेवा प्रबंधक से उपयोगकर्ता तक। वे अपने साथ जिम्मेदारियां भी स्थानांतरित करते हैं। इन तीनों में से किसी के बारे में ईमानदारी से बात करने के लिए अन्य दो के बारे में भी बात करना आवश्यक है।*

## स्रोत और आगे पढ़ने के लिए


- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, 2013 का Bitcoin सुधार प्रस्ताव। क्रिप्टो उद्योग में रिकवरी वाक्यांशों के लिए वास्तविक मानक।
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), जिसमें Ed25519 शामिल है। IETF, जनवरी 2017। समकालीन उद्योग के बड़े हिस्से में उपयोग की जाने वाली हस्ताक्षर योजना का मानक विनिर्देश।
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, संस्करण 2.0। IETF, सितंबर 2000। वाक्यांश से बीज (seed) व्युत्पन्न में उपयोग किए जाने वाले PBKDF2 एल्गोरिथम को परिभाषित करता है।
- विनियम (EU) 910/2014 (eIDAS) और विनियमन (EU) 2024/1183 (eIDAS 2) द्वारा इसका विकास — इलेक्ट्रॉनिक पहचान और योग्य हस्ताक्षर के लिए यूरोपीय ढांचा। स्व-संप्रभु से अलग शासन, लेकिन वैचारिक रूप से उन्हीं क्रिप्टोग्राफ़िक प्रिमिटिव द्वारा समर्थित।
- Allen, C. — *The Path to Self-Sovereign Identity* (2016)। स्व-संप्रभु मॉडल के सिद्धांतों और प्रतिबद्धताओं पर विहित पाठ, पहले का लेकिन समकालीन समाधानों के परिवार को समझने के लिए प्रासंगिक।



← पिछलाभरोसे के संकेत के रूप में बिजनेस मॉडलअगला →पेशेवर अभ्यास के रूप में सेल्फ-होस्टिंग


## हाल की रीडिंग


- विचार · 29 जून 2026 आप अनाम नहीं हैं
- चिंतन · 27 मई 2026 कुछ ऐसा जिसे एक हस्ताक्षर ठीक नहीं कर सकता
- विश्लेषण · 26 मई 2026 वास्तविक बनाम आभासी गोपनीयता: वे प्रश्न जिन्हें आपको खुद से पूछना चाहिए


इस लेख को अपने साथ ले जाएं जहाँ भी आपको इसकी आवश्यकता हो।

↓ मार्कडाउन ↓ सादा टेक्स्ट ↓ PDF

फ़ाइल आपके डिवाइस पर डाउनलोड हो जाएगी। वहां से आप इसे सहेज सकते हैं, Solo2 में आयात कर सकते हैं या जहां चाहें साझा कर सकते हैं। Cuadernos आपके लिए गंतव्य तय नहीं करता है।


मोहरबंद · SHA-256 28ea504c50476c7ab75797ec093cc63dc750b89e3d812f44b99082804a834d77

Cuadernos Lacre · Menzuri Gestión S.L. का एक प्रकाशन ·
R.Eugenio द्वारा लिखित · Solo2 की टीम द्वारा संपादित।

यह वेबसाइट कुकीज़ का उपयोग नहीं करती है। आपका ब्राउज़र जो कुछ भी लोड करता है वह हमारे द्वारा लिखा या पर्यवेक्षित है और हमारे यूरोपीय सर्वरों पर होस्ट किया गया है: अनाम विज़िट काउंटर (Umami, स्व-होस्ट किया गया) और भाषा चयनकर्ता तथा आपकी लाइट/डार्क थीम पसंद के लिए आवश्यक न्यूनतम जावास्क्रिप्ट, जो आपके अपने डिवाइस पर सहेजी जाती है। बाहरी कंपनियों के कोई संसाधन नहीं, कोई ट्रैकर नहीं, कोई प्रोफाइलिंग नहीं, कोई डेटा साझाकरण नहीं। यदि आप हमें फ़ॉलो करना चाहते हैं: RSS।